Lesson learnt from Mini2:

1.The most important part I have learnt during Mini2 is team cooperation. We have to design together, code together and share ideas. We have to keep pace with each other and communication often to push the whole project moves forward. The whole project not only improves my coding skills, but also be more aware of the importance of cooperation.

2.Learn how to write documents. In Mini2, I drawed class diagram and flowcharts. I wrote and clear up the design presentation and final presentation documents and ppt. I also wrote use case and I learnt how to write user story and draw page flow.

3.Learn a new software -- android studio. It is a powerful tool for android developer. I can design UI, write core code, build gradle and connect github by using it.

4.Learn how to implement android application by android studio. Although we starts from easy and simple application, they are essential part for us to get familiar with the develop environment and process.

5.Know the difference between create android project with and without an activity. An activity is essentially a piece of UI component, typically corresponding to one screen. Almost all activities interact with the user, such as dial the phone, take a photo, or view a map. Activities can present to user as full-screen, floating windows and embedded inside of another activity. One application has multiple activities, but only one “main” activity. Since activity is the primary mechanism for interacting with the user, create android project without an activity means no interaction between the user and system. Service class can be used to create android project without activity. It can perform a long-running operation while not interacting with the user or to supply functionality for other applications to use.

6.Have a deep thinking of the future of mobile computing. I watched the video professor provided and was touched deeply. My conclusion is: the development of software, hardware and new user experience will make mobile computing more popular and occupy more marketing shares than desktop computing.

7.Know the lifecycle of activity. Each activity can start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. The back stack abides to the basic "last in, first out" stack mechanism, so, when the user is done with the current activity and presses the Back button, it is popped from the stack (and destroyed) and the previous activity resumes. When an activity is stopped because a new activity starts, it is notified of this change in state through the activity's lifecycle callback methods. There are several callback methods and each callback provides you the opportunity to perform specific work that's appropriate to that state change. For instance, when stopped, your activity should release any large objects, such as network or database connections. When the activity resumes, you can re-acquire the necessary resources and resume actions that were interrupted. These state transitions are all part of the activity lifecycle.

8.Know the lifecycle of android application. Applications run in their own processes. Processes are started and stopped as needed to run an application's components. Processes may be killed to reclaim resources. The end of android application is not determined by itself, it is decided by android system. Nowadays, most of smartphones supports multi-tasking. If the ending of application is decided by itself, there is a serious problem about memory management. For example, a user keeps opening applications. As the number of opening applications grows larger, system memory might be consumed quickly and the system will die. To avoid that terrible case, the lifecycle of android application is determined by android system. The advantage is when memory is not enough, system will end some trivial applications to recycle memory for the usage by more important applications.
The three lives of android:
1>.The Entire Lifetime: the period between the first call to onCreate() to a single final call onDestroy().
2>.The Visible Lifetime: the period between a call to onStart() until a corresponding call to onStop().
3>.The Foreground Lifetime: The period between a call to onResume() until a corresponding call to onPause().

9.Learn how to use intent to jump between activities. I just need to set target activity and startActivity. I can also use intent.putExtra() and getExtra() get set and get values.

10.Learn how to use fragment and jump between fragments. Use fragment properly is a good design habit. I once encounter a problem when I need to jump from one fragment to another one and then return back. The solution is to setTargetFragment first.

11.Learn how to use content provider. SQLite is a local database that I can use to store information in my application. It is convenient and useful.

12.Learn how to design android application UI. To be honest, it is very tricky when I first use android layout. But after then, I learnt how to design UI by writing code. In that way, I can use all kinds of layout and put all components as I want.

13.Learn how to play music and video in android studio. The class to use is MediaPlayer. I learnt how to play music and video from local or internet.

14.Learn how to record audio and record the uri in Amazon S3.

15.Learn how to take photo by camera or select photo from photo album, and then record the uri in Amazon S3.

16.Get familiar with google map v2 api. I learnt how to set Marker and get location, add my location button, and update camera.

17.Exception handling. As this course ending, my intuition to consider all kinds of exception and handle them always comes first when I create android application. For the final project, I deal with all types exception in user home page and issue activity page. I checked if the input is empty or invalid. If it is, there is a toast text to notify user to rectify it.

18. Material Design has been adopted in the our project. The new features of Android Design make the layout of application look more beautiful and user-friendly.

19. Re-Usage of code fragments. As considerable of functions are duplicated, codes can be formulated and compacted into modules, which can be used by several different parts of the applications. For example, layouts of some recyclerview items.

20. Communication between android devices and desktop server. Socket communication technology is applied in the project. By setting multithreading server in a desktop, server can handle requests from various android devices simultaneously.

21. Noted that in order to make socket communication successful, a. name of packages involved in socket communication should stay the same in server and client; b. objects of class should be declared serializable; c. serial number ought to be added into class.

22. Child classes of AsyncTask have been widely used in our application. This provides a platform for simple realization of running tasks in background. For example, downloading/uploading images from/to cloud server.

23. When same fragments are to be called multiple times in the same activity, its layout should not be inflated more than one time. In order to avoid this issue, the root view inflated from the fragment layout should be declared as static.

24. Design of database should conform to Normalization Requirement. Large amount of data is needed to be stored in server.This principle helps reduce the data stored in database and makes better use of memory. It also accelerated the speed of searching.

25. When data is populated from database or to stored within database, it should be stored within objects and passed to client. Such implementation not only accords with the idea of object-orientation programming, it also makes code look more precise and clean.

26. Classes should be designed as self-contained. CRUD methods ought to be added into entity classes for each field. This design helps manipulate objects more easily.

27. When socket communication is to be implemented, a new thread should be created to handle the implementation. Application activity should mainly focus on layout, not background task.

28. Constant value in layout should be declared within values xml files, not hard coded within layout file directly. This makes manipulation and management of values in a more systematic way and get rid of project warning.

29. Design of data file structure hierarchy ought to accord with functionalities of classes stored inside.   Meaningful names given to each package (based on functions performed by the class group) improves readability of code.

30. Static declaration of keys of extras, the data to be communicated between activities or fragments, helps eliminate possible typing bugs in code. Also, meaningful names given to keys provides useful debug information.

31. Fragments contained within fragment ought to be avoided. Improper handling of fragment cycles, such as re-inflate of fragment layout before destruction of the same fragment will render serious system error and makes application abruptly shut down.

32. In order to make the multi-support implementation of devices more easily, dimension of each view inside the layout, especially relativelayout, can be declared within dimen.xml files for different screen sizes.

33. Useful information should be added adequately within codes and printed out on screen or within console for different purpose. Information printed on device screen, eg. using Toast, gives user a clear mind of the running status of the application. Information given in console, however, helps developer locates abnormal situations faster and solve them more easily.

34. Invocation of activity should be avoided as far as possible. Because existences of activities may still take up memory even though they are out of user sight. Instead, transaction of fragments can better reduce usage of memory and efficiently save power than switching between activities.

35. Learn how to use MySQL in an Android App. And several methods can be used in it such as creating, reading, updating and deleting. All the methods are typed in text file and parsed in java language.

36. There are always bugs about database, maybe MySQL, maybe SQLite. Sometimes is null pointer exception, sometimes is out of bound exception. There may be because the database isn’t cleared well. So uninstall the app or delete tables in the database may be a good help.

37. Android studio is a good software for designing Apps. But it has some difference with eclipse, such as it can not get the local address by methods, or it doesn’t have the button to stop the project.

38. In android studio, I learnt how to design UI for my App. I learn how to use textView, editText, input space, button, navigation bar and so many others that I have only seen but did not how to design it before.

39. In our team project, we save the user information and event information in remote database using MySQL. It can be achieved from our app to the server. We save chats information in local database using SQLite. We can get chatting records even off line. Through some tests, the database is saved on each application, which means after deleting the software, the database is deleted then.

40. In our database design, we just add each chats information and message information in it but ignoring it belongs to whom, which is one shortcoming. It may cause problems when several users log in on one device one by one. It can be solved by creating several databases, and one for one user. We want to make it true it time permits.

41. We learn how to chat in our application. It needs a service in the cloud to receive the message that are sent to it and send them to the receiver.

42. For chatting with other users, we need to use a service on the internet.So we will add some jar files to the library. It needs a key and password set in our App. And what’s more, it also needs to be stated in manifest.

43. In our application, we can make it true that when pressed on a button, it looks darker than normal. It can be realized by adding a xml file that setting two pic when the state_pressed is true or false.

44. Different API are designed are different processor architectures. Some are for Arm and some are for X86. But we can add several files in jniLibs file in src/main/ to make our application used in every architectures.

45. We learn how to use process dialogue in android studio. It can be used when we want to receive some signals. Such as in our application, we want to wait for the service being connected. So we open the process dialog and create a listener. If it has received the signal, the dialog can dismiss.

46. Although we create a database in our application, but there are some websites that can help us with database and server. We can save what may be used in the cloud and get them when we want. But in our application we do not use it as we can examine our database when testing and debugging.

47. Like we can create an intent, put some values in it to send and start a new activity with this intent. We can also start a service with intent and method of startService. We can also send values with putExtra method and receive them at the other side.

48. We learn how to use listView. We can use an arrayAdapter to fill in the listView. What’s more, we can define an adapter class and get what we need and make them different views and lookings.

49. When we meet some errors and debug, we can put something in the log, and see what it is and whether it is the same with what we consider it to be. This may produce great help when we can not find where exist error.

50. From this application testing and debugging experience, the exception stack printer only tells us where it causes error or exception but not where the code is wrong. So we learn how to use the information it offers to debug our code.